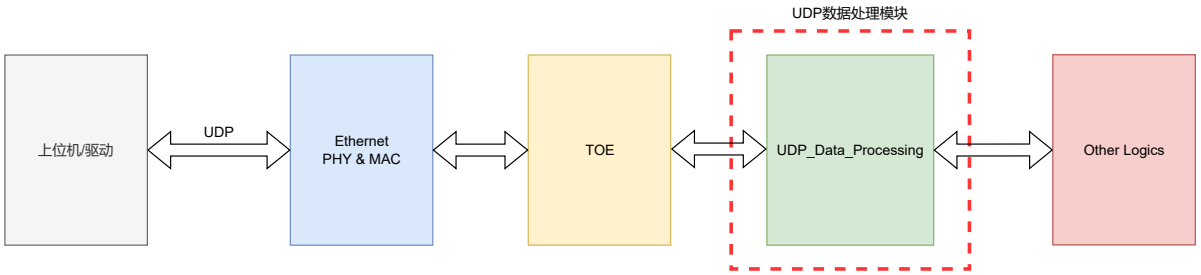
功能描述

1. 用途

该模块负责对接收的UDP数据包进行解析，并实现对应的功能输出；同时也负责将来自外部模块的数据封装为特定格式，发送给TOE，完成UDP数据发送。

2. 外部信号流



3. 模块接口

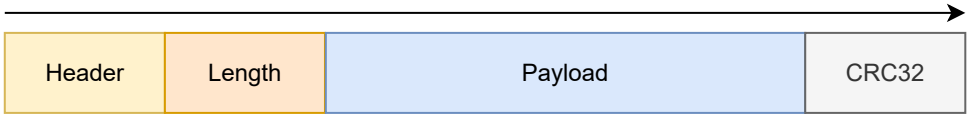
序号	信号名称	位宽	输出/输出	描述
1	clk	1	I	时钟
2	reset	1	I	高电平复位
3	eth_linked	1	I	信号为高时，表示网络连接成功
4	pulse_second	1	O	秒脉冲输出
5	udp_app_recv_fifo_rden	1	O	UDP数据接收端口，FIFO读使能，同时也是udp_app_recv_fifo_rddata的有效信号
6	udp_app_recv_fifo_rddata	8	I	UDP数据接收端口，接收到的UDP数据
7	udp_app_recv_fifo_rdcnt	12	I	UDP数据接收端口，FIFO读计数
8	udp_app_recv_fifo_empty	1	I	UDP数据接收端口，FIFO空标志
9	udp_app_send_fifo_rden	1	I	UDP数据发送端口，FIFO读使能，同时也是udp_app_send_fifo_rddata的有效信号
10	udp_app_send_fifo_rddata	8	O	UDP数据发送端口，发送到的UDP数据
11	udp_app_send_fifo_rdcnt	12	O	UDP数据发送端口，FIFO读计数
12	udp_app_send_fifo_empty	1	O	UDP数据发送端口，FIFO空标志
13	TOE_reg_wren	1	O	TOE寄存器接口，写命令有效信号
14	TOE_reg_rden	1	O	TOE寄存器接口，读命令有效信号
15	TOE_reg_din	66	O	TOE寄存器接口，读/写命令数据
16	TOE_reg_out_valid	1	I	TOE寄存器接口，TOE返回数据有效信号
17	TOE_reg_dout	66	I	TOE寄存器接口，TOE返回数据
18	CMU_Status_Info_ready	1	O	通信板状态信息接口，ready
19	CMU_Status_Info_valid	1	I	通信板状态信息接口，valid

序号	信号名称	位宽	输出/输出	描述
20	CMU_Status_Info_last	1	I	通信板状态信息接口，last
21	CMU_Status_Info_data	32	I	通信板状态信息接口，data
22	XYZ_Status_Info_ready	1	O	调控板状态信息接口，ready
23	XYZ_Status_Info_valid	1	I	调控板状态信息接口，valid
24	XYZ_Status_Info_last	1	I	调控板状态信息接口，last
25	XYZ_Status_Info_data	32	I	调控板状态信息接口，data
26	DAQ_Status_Info_ready	1	O	读出板状态信息接口，ready
27	DAQ_Status_Info_valid	1	I	读出板状态信息接口，valid
28	DAQ_Status_Info_last	1	I	读出板状态信息接口，last
29	DAQ_Status_Info_data	32	I	读出板状态信息接口，data

4. 数据协议

UDP接口的数据协议如下所示。

比特序	字段	长度/bit	描述
[31:16]	Header	16	数据包头；0x4d44，控制命令；0x4547，TOE寄存器命令；0x494e，状态信息。
[15:0]	Length	16	Payload字段的长度，单位为byte
/	Payload	32*N	负载数据，长度为32-bits的整数倍
[31:0]	Check	32	CRC32校验，计算Payload字段的CRC32



CRC32的生成多项式为：$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$，初始值为0xffffffff。

Payload共有3中类型：控制命令，TOE寄存器命令，状态信息；通过Header区分不同的类型。
0x494e 对于TOE寄存器命令，每条命令为64-bits，如下表所示；一个Payload里可以有多条命令。

比特序	字段	长度/bit	描述
[63:56]	RSV	8	
[55:52]	SFP ID	4	取值范围：0x0，0x1，0x2，0x3

比特序	字段	长度/bit	描述
[51:49]	RSV	3	
[48]	读写标志位	1	write：0；read：1
[47:32]	TOE寄存器地址	16	
[31:0]	TOE寄存器数据	32	

5. 模块功能

5.1 UDP数据帧解析

UDP数据通过UDP数据接收端口（udp_app_recv_fifo_xx信号）输入到模块，根据章节4描述的数据协议，区分出不同的数据类型、提取Payload、进行CRC校验。

5.1.1 控制命令

数据包头；0x4d44，**控制命令；0x4547**，TOE寄存器命令；0x494e，状态信息。

控制命令数据帧的header为0x55434d44，目前支持状态上报开关命令，后续根据需求增加命令。

命令	Payload	描述
开启状态上报	0x1000_0001	
关闭状态上报	0x1000_0000	

开启状态上报后，允许本模块通过UDP发送状态数据，同时本模块会输出一个秒脉冲信号（每秒钟输出一次，脉冲宽度为16个时钟周期）给到外部；关闭状态上报后，不允许本模块通过UDP发送状态数据，也不会输出秒脉冲信号。

5.1.2 TOE寄存器命令

数据包头；0x4d44，控制命令；0x4547，**TOE寄存器命令；0x494e**，状态信息

Header为0x55524547，上位机/驱动通过TOE寄存器命令来读写TOE寄存器，每条命令为64-bits，如章节4所描述。本模块将解析到的TOE寄存器命令通过TOE寄存器接口发送到外部TOE模块，其中TOE_reg_din的数据结构如下所示，TOE_reg_wren拉高表示为寄存器写命令，TOE_reg_rden拉高表示为寄存器读命令。

比特序	字段	长度/bit
[65:64]	SFP ID	2
[63:32]	TOE寄存器地址	32
[31:0]	TOE寄存器数据	32

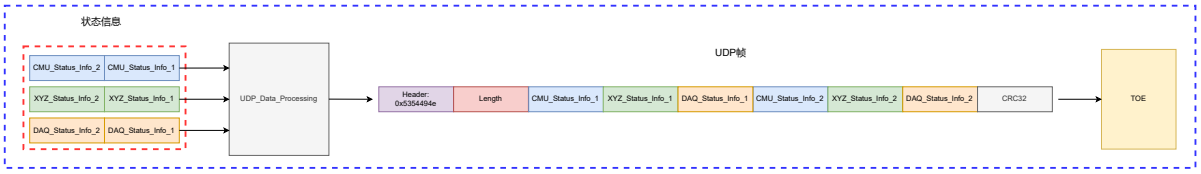
TOE模块在接收到TOE读命令后，会返回寄存器的读出数据到本模块（TOE_reg_out_valid和TOE_reg_dout）。返回的TOE数据也满足上表的数据结构，本模块需要将返回的TOE数据封装为章节4中的UDP数据帧，再通过UDP数据发送接口发送数据。

5.1.3 CRC校验

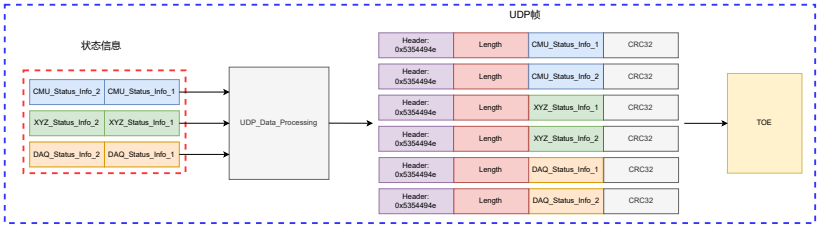
对于接收到的每个数据帧，根据章节4的描述计算Payload字段的CRC32，并与数据帧末尾的CRC32比较，如果不一致，产生一个校验错误信号（一个时钟周期高电平）。

5.2 状态信息上报

本模块从其他模块接收3种状态信息：通信板状态（CMU_Status_Info）、调控板状态（XYZ_Status_Info）和读出板状态（DAQ_Status_Info），本模块将接收到的这些状态信息组帧后发送到UDP数据发送端口（udp_app_send_fifo_xx信号）。
状态信息也是以数据帧的形式输入到本模块，xx_last即为帧结束指示，数据帧的时序满足AXI-Stream协议。本模块将一个或多个状态信息帧组成一个Payload，然后封装为UDP帧发送到UDP数据发送端口。一个状态信息帧是不能被拆分的，不能将一个状态信息帧分到多个UDP帧中。下面是两种可能的组帧方式的示例。



多个状态信息帧组成一个Payload



一个状态信息帧组成一个Payload

状态信息都满足tansaction的数据结构，如下图所示。可以直接提取transction中的LENGTH字段，来计算UDP帧的长度，不需要缓存整个状态信息帧再来计算长度。

序号	数据结构		长度	比特序	功能描述	备注
1	CMD	W/R	1bit	[63]	读写标识，0：W；1：R	
		ARD FLAG	1bit	[62]	主动回读数据标志	主动回读模块根据读出芯片读请求读出实验数据
		CID	5bit	[61:57]	芯片组中每个芯片的ID标识	包括量子测控芯片组及其他需要配置的芯片组
		ADDR	25bit	[56:32]	低25位地址	可与EXADDR拼接使用，或直接映射到量子测控芯片的偏移地址
		EXADDR	12bit	[31:20]	扩展地址，可根据需求划分为多个功能区（4比特预留+5比特槽位+3比特板卡内部预留）	可与ADDR拼接使用此时寻址空间为128GByte； 该4096K的地址空间可按照需求映射给集控中心和外部SIP芯片组
		LENGTH	20bit	[19:0]	指示读写数据长度，以Byte为单位	最大支持1MByte
2	DATA		N*4 Byte	[31:0]	数据	读操作时，无需包含该项

实现过程

1.用途拆分

1.TOE下发的数据，要能够分发执行。即解析与执行。

2.把各个模块的数据收集一下，交给TOE，然后上传到MAC/PHY。即接收与转发

2.解析与执行的设计分析

TOE的UDP端口和这个UDP_app是用FIFO形式的结构去通信的，我们拿到payload要先判断是3类信息中的哪一类。

如果是TOE寄存器命令那就要按照结构来解析了。解析出来后按照66bit的结构发送出去，去控制TOE。

如果header为0x55434d44 ，就为控制命令 ，相当于一个开关，能够打开和关闭秒脉冲。

状态信息是要上报的，所以要接收过来的状态信息。和众多板子的通信是一种简化板的axi_stream来通信的，叫信息接口，因为不需要控制板卡，只需要上传信息给toe所以用四根线就行了。

2.1解析状态机

以32bit为单位，成功读出数据。udp_recv_data[31:0]　udp_recv_data_tvalid配合。

接下来要设计一个状态机解析获取的data。

P_IDLE: 如果没有检测到udp_recv_data_tvalid ，就一直循环，不断拉低recv_udp_frame_last。 如果检测到了，才能进入下一个状态。

这个状态能获取第一个数据，得到header_reg，payload_remain信息。

P_DATA：没有检测到udp_recv_data_tvalid ，就一直等在这个状态，啥也不做。 检测到了进入数据处理： 用data_reg缓存当前数据。payload_remain指示未缓存的数据还有多少个。 当所有payload 都被data_reg缓存完了（payload_remain ==0），才能进入下一个状态。先不断检测余量，余量为0直接跳到空闲态，需要注意的是检测的余量为0的时钟周期，crc32_recv数据到了，需要接收一下。如果余量不为0，才继续缓存，缓存一次后更新余量。

数据的完整周期结束。

2.2观察解析状态机仿真

数据源 udp_app_recv_fifo_rden　udp_recv_data_reorder[31:0]　udp_recv_data_tvalid

帧状态判断的： parse_state[1:0]　payload_remain　recv_udp_frame_last

CRC校验：crc32_calc[31:0] crc32_recv[31:0]（慢一个时钟，早实际上收到了）　crc_match

存数据的： data_fifo_wren　data_fifo_din[31:0]

我们需要等一整个帧结束 验证了正确性后，才能对数据进行处理。

recv_udp_frame_last 为1代表收到了完整一帧。

完成验证。

crc_match 拉高，代表收到了完整正确的一帧。

接下来进行数据处理。对于控制命令来说，就只有32bit的数据。0x494e 对于TOE寄存器命令，每条命令为64-bits，一个Payload里可以有多条命令。

先用data_fifo存数据。存数据会存IDLE和DATA , CRC不存。 存好完整一帧之后，要进行处理。处理完成之后，才能运行下一帧进来。

2.3处理状态机

处理和执行又需要处理状态机来完成。和上边不同的是，上面的是从不确定的线中去数据，数据不一定正确。下面的状态机是从FIFO中取数据，状态是一定正确的，下面的就不需要CRC校验了。 关键的是，底下的我想什么时候取就什么时候取，不需要等tvalid信号。

D_IDLE：不断去检测 recv_udp_frame_last。如果拉高 ，就把deal_busy拉高，进入处理状态。然后拉高data_fifo_rden，保证下一个时钟可以获取头和长度信息。拉高后之后进入下一个状态。

D_HEAD：判断头的类型。如果是TOE头，我们需要接收两次数据才能操作一次，择跳转到TOE_1状态。

D_TOE:D_TOE_1获取地址信息之后，跳转到TOE_2获取数据信息 ，之后跳转到D_EXE。

D_EXE：如果判断是TOE头，会执行用上述TOE缓存执行TOE操作。执行完判读是否非空，非空的话再跳转到TOE_1继续执行，此时data_fifo_rden是0状态，在TOE_1阶段，data_fifo_dout不会是前32bit。

2.4观察处理状态机仿真

状态：deal_state[2:0] deal_busy

取数据 data_fifo_rden data_fifo_dout[31:0] data_fifo_empty

TOE_SFP_ID[3:0] TOE_RW_FLAG TOE_ADDR[31:0] TOE_ADDR[31:0]

卡住了，花了很多时间

读FIFO 时：header_reg <= data_fifo_dout[31:16]; 赋值不进去。FIFO读出时钟有问题。时钟给我搞晕了。先拉高，再触发。不能拉高的时候触发。 还有读empty拉高疑问，要先读出来，要在后一个时钟才能用。

2.4.1TOE命令执行

通过这三个来操作

```
output    TOE_reg_wren,
output    TOE_reg_rden,
output  [65:0]  TOE_reg_din,
```

观察 这三个 +数据缓存 仿真正确。

梳理一下

我从udp_app_recv_fifo 拿到数据，如果是TOE类型的数据，解析成 TOE_reg_wren TOE_reg_rden TOE_reg_din输出。

我从 TOE_reg_dout TOE_reg_out_valid 拿到TOE 数据，包装为下面形式，再通过udp_app_send_fifo发送

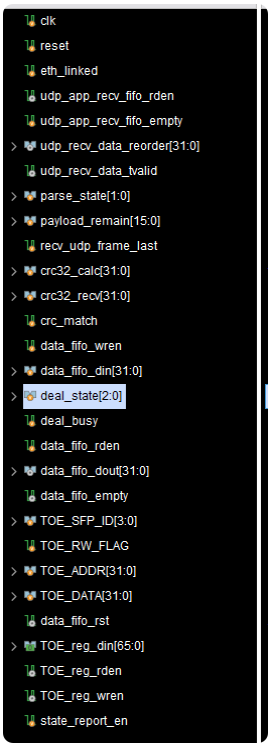
比特序	字段	长度/bit	描述
[63:56]	RSV	8	
[55:52]	SFP ID	4	取值范围：0x0，0x1，0x2，0x3
[51:49]	RSV	3	
[48]	读写标志位	1	write：0；read：1
[47:32]	TOE寄存器地址	16	
[31:0]	TOE寄存器数据	32	

比特序	字段	长度/bit
[65:64]	SFP ID	2
[63:32]	TOE寄存器地址	32
[31:0]	TOE寄存器数据	32

2.4.2CTRL命令执行

执行完成。deal_state[2:0] 从 过程是 0 1 5 015

1状态要判断头 5状态要判断payload。 完成执行命令，存档



秒脉冲完成， crc错误信号完成

3.接收与转发的设计分析

3.1UDP转发

UDP发送FIFO接口

input udp_app_send_fifo_rden,

output [7:0] udp_app_send_fifo_rddata,

output [11:0] udp_app_send_fifo_rdcnt,

output   udp_app_send_fifo_empty,

梳理分析

我们搞一个FIFO 输出口就是上面的。然后只需要往里边填数据就行了。

TOE_reg_out_valid 和TOE_reg_out简单，把valid但做写使能就行。

s_axis_t的三根线和 FIFO可以无缝衔接。

TOE valid的数据需要用fifo接收 65 to65 。

需要有一个仲裁模块。还需要一个控制 CMU_Status_Info_ready 等3个信号的模块。

3.2仲裁与存储状态机

IDLE： 不断检测是否有TOE非空信号或valid信号过来。有的话进入对应处理机制 。

WAIT_READ

RECV_TOE：收到TOE的66bit数，直接按结构重新组帧。

RECV_CMU：将完整帧的数存入FIFO中

3.3发送状态机

要把TOE或板子接收FIFO中的数读出来再发送出去，发送也就是往fifo_wr_en fifo_wr_data中写入数据

SEND_IDLE

SEND_TOE_HEAD_LENGTH

SEND_TOE_DATA_1

SEND_TOE_DATA_2

SEND_CRC

SEND_INFO_HEAD_LENGTH

SEND_INFO_DATA

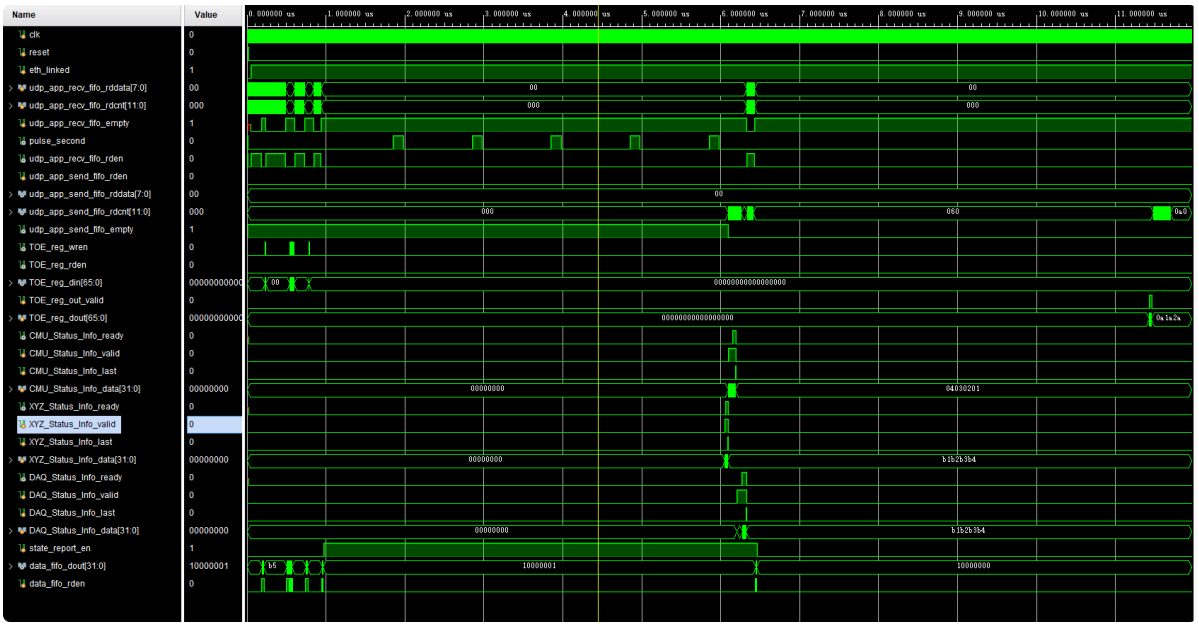
3.4板子缓冲FIFO

板子的缓冲fifo叫：info_cache_fifo。

对于仲裁储存状态机：要把数据写入缓冲FIFO 。写入完成。

发送状态机。读缓冲FIFO的数。

全部完成，进入仿真状态。



验证过程

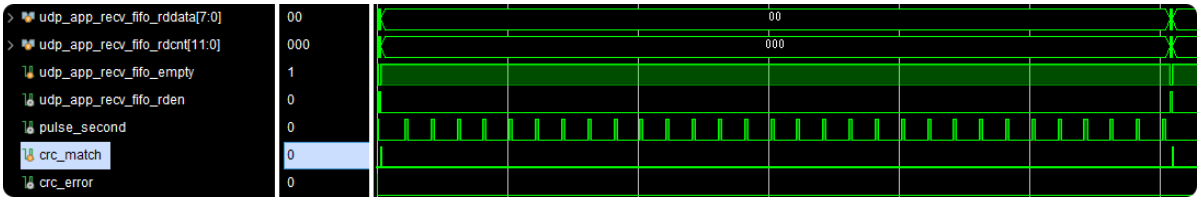
1.解析与执行验证

1.1秒脉冲功能

发送指令开启和关闭。

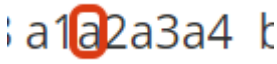
```
send_fifo_data(64'h4547000410000001);
send_fifo_data_32(31'h34D1ED6F);

#30000;
// send_xyz_data( 32'hbb020304, 32'h0
// send_cmu_data( 32'haa020304, 32'h0
// send_daq_data( 32'hcc020304, 32'h0
send_fifo_data(64'h4547000410000000);
send_fifo_data_32(31'h3010F0D8);
```

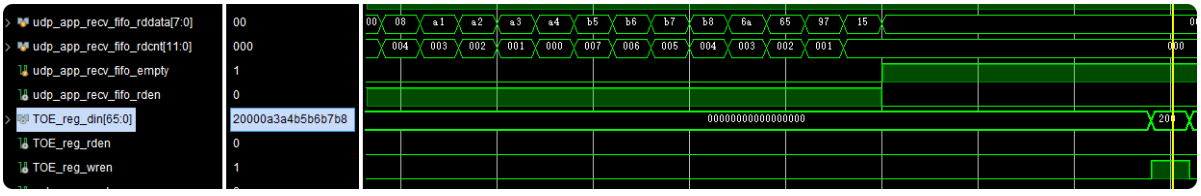
能够正确执行秒脉冲功能。CRC校验码正确。



1.2TOE控制执行

在仿真中，朝着recv端口放一条UDP结构的TOE指令：494e 0008 a1a2a3a4 b5b6b7b8 6A659715 中间的a1a2a3a4 b5b6b7b8就是TOE负载。 地址是a3a4 数据是b5b6b7b8 。SFP是第53-52位，也就是a=1010中后面的10，也就是2。



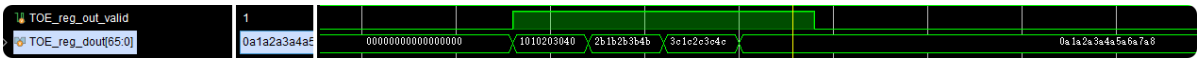


模块能很好地执行TOE控制。可以看到写入了一个2 0000a3a4 b5b6b7b8 SFP是2，地址是a3a4 数据是b5b6b7b8，和上面的，一致功能正确。

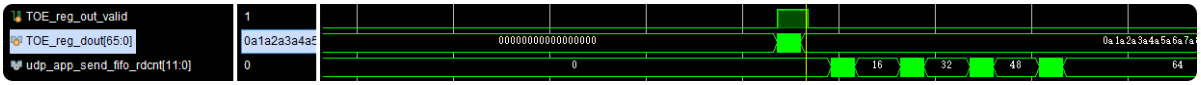
2.接收与转发验证

2.1TOE转发功能

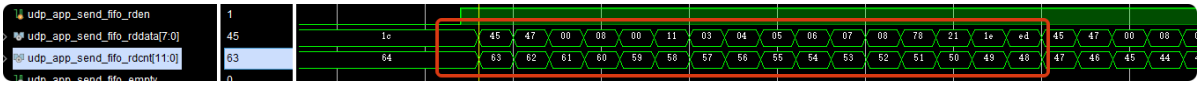
TOE_reg_out输出了4条命令，模块将其解析重组后，进入发送FIFO中。

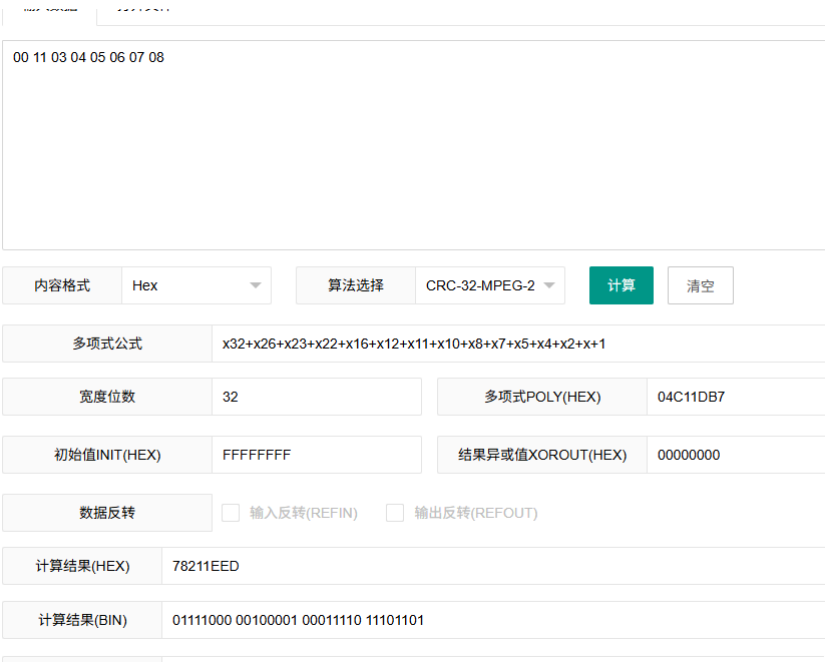


发送FIFO中的计数值为16，32，48，64，说明一条8字节的命令被包装为16字节（头4字节+命令8字节+CRC码4字节），数据量完整。

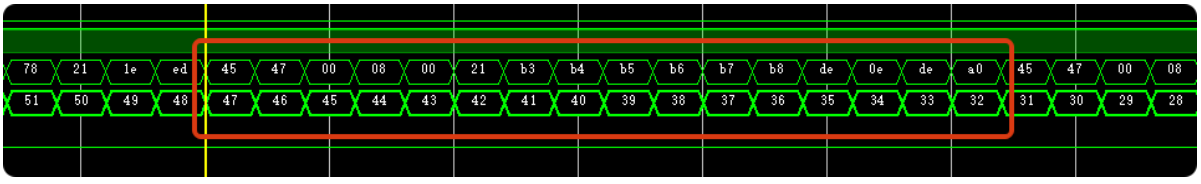


拉高fifo_rden将其读出，可以看到第一条头为4547，长度为8字节，为读标志位，SFP为1，校验码为78211EED（正确），数据内容与原帧一致。

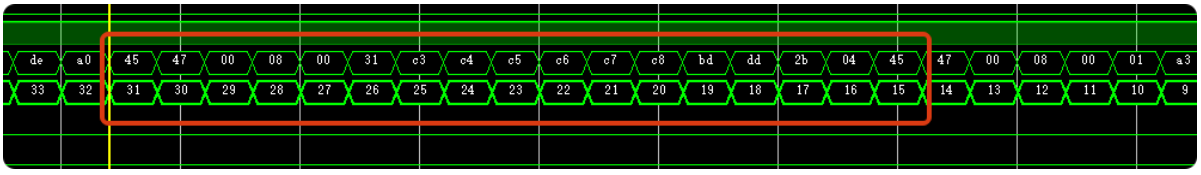




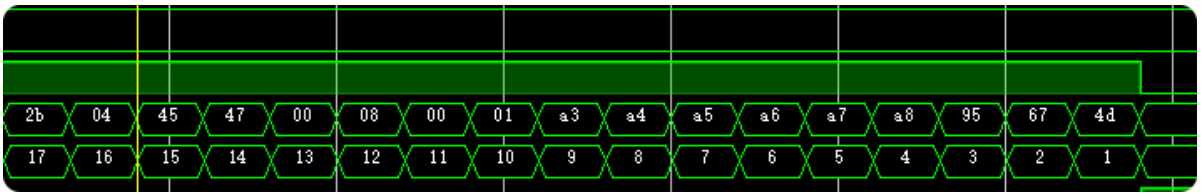
第二条，同理分析，无误。



第三条，无误。

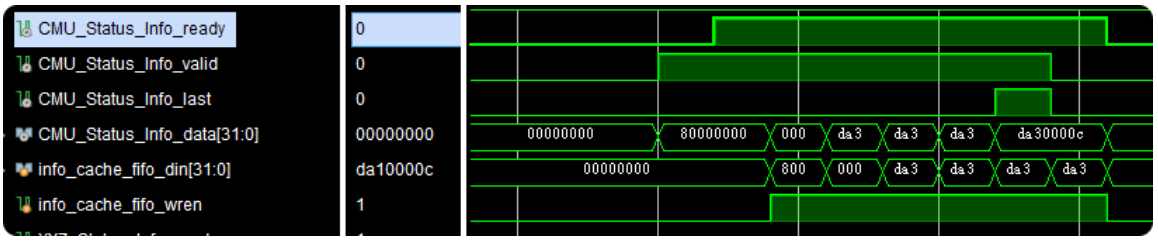


最后一条无误。TOE命令的接收与转发功能正常。

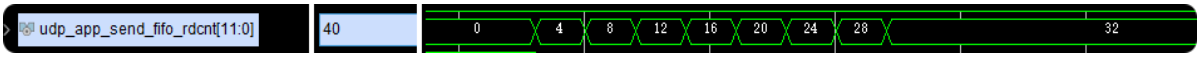


2.2状态信息转发验证

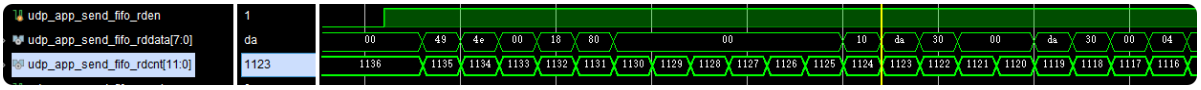
接收到状态信息为6x4=24个字节　（指令头8字节加上4个4字节的数据）



写入发送fifo为32个字节（24字节加上8字节的head和crc），数据正确



读出发送fifo的内容验证正确。494e是包头，0x0018是长度。0x18的十进制数是24，对应接收到的24字节的状态信息。



读出数据的头计数值是1135，尾巴的计数值是1104，一共有32个数据。也就是32个字节，符合整帧32个字节。

2.2.1CRC校验码验证

通过在线完成和最后输出结果进行对比，结果一致

CRC在线计算

标签 后端 硬件开发 摘要

输入数据 打开文件

80000000 00000010 da300000 da300004 da300008 da30000c

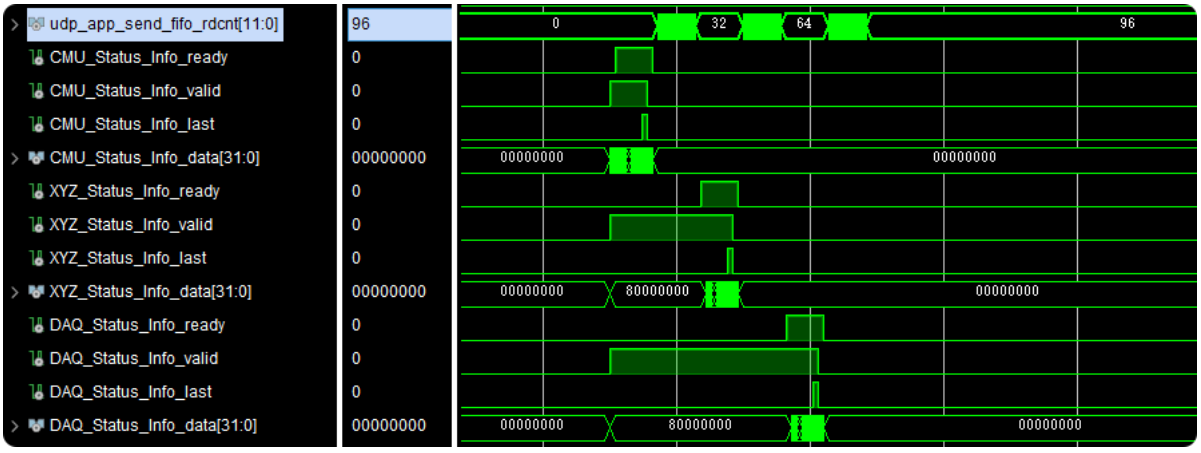
内容格式	Hex	算法选择	CRC-32-MPEG-2

计算 清空

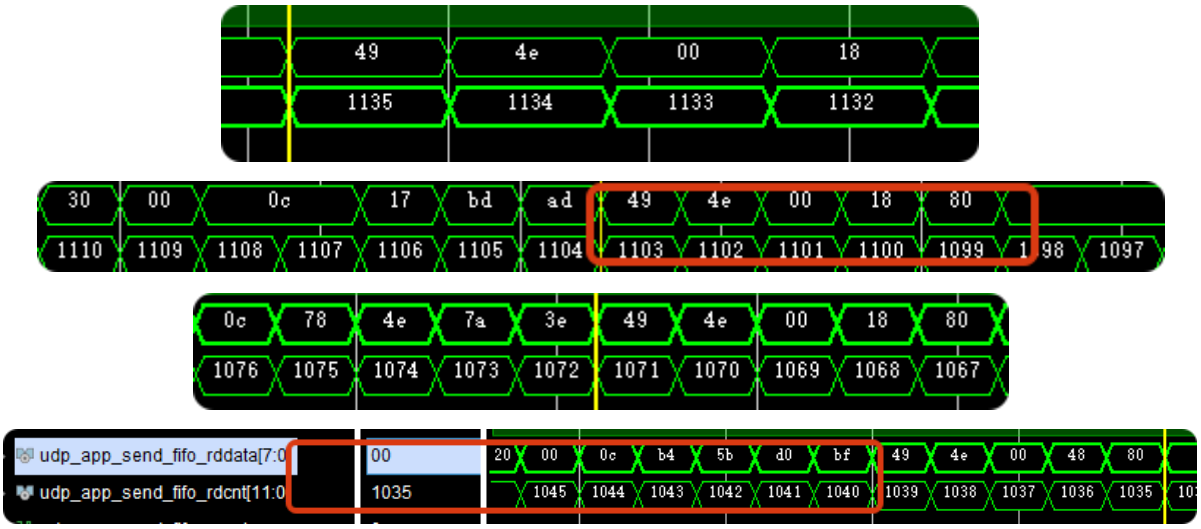
多项式公式	x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x+1
宽度位数	32
多项式POLY(HEX)	04C11DB7
初始值INIT(HEX)	FFFFFFFF
结果异或值XOROUT(HEX)	00000000
数据反转	输入反转(REFIN) 输出反转(REFOUT)
计算结果(HEX)	0C17BDAD
计算结果(BIN)	00001100 00010111 10111101 10101101

2.2.2多模块联合验证

3个模块valid同时拉高，模块会先读CMU，再读XYZ，再读DAQ,与仲裁选择优先级一致，正确。可以到FIFO中的值从32到64再到96,说明3个帧数据(包括头和CRC)已经完整地写入了FIFO。后面的72+8长度的帧也验证正确了和这个一样。



读FIFO验证数据第一个数据头在1135，第二个头在1103，第三个头在1071 ，尾巴在1040. 说明这3个帧数据均为32Bytes，数据完整。

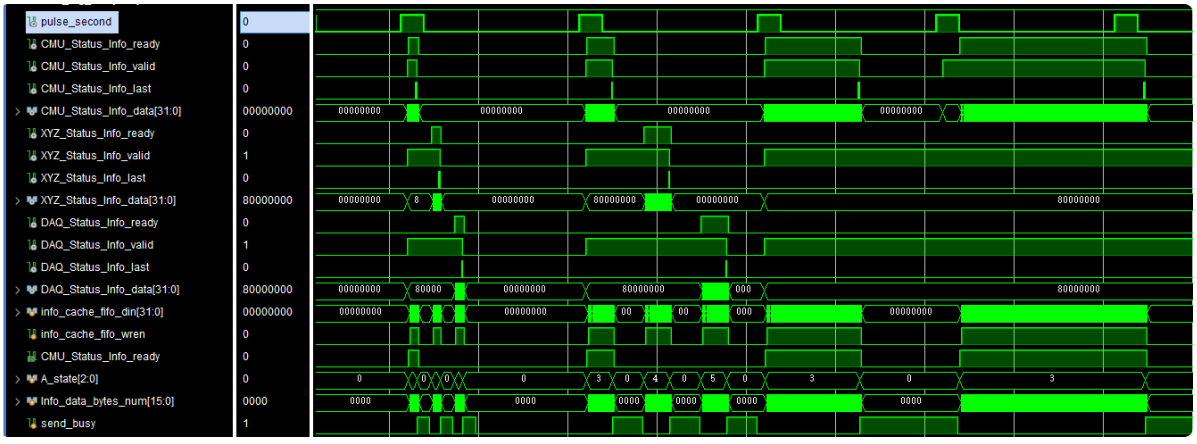


2.2.3数据率不匹配的情况

因为只有一个数据输出口，四个数据输入口（TOE输入、XYZ、CMU、DAQ输入）。 理论极限情况下，输出的速度(32bit x125MHz)只有输入数据(4 x 32bit x125MHz)的1/4，这是无法改变的带宽限制。 如果一个秒脉冲下，多级输入数据速度超过输出数据的速度，就会出现数据积累的情况，无法避免。 虽然会累积，但是数据不会丢失，会按一定优先级发送。数据积累情况下会优先发送cmu帧，接着发xyz帧,最后daq帧。

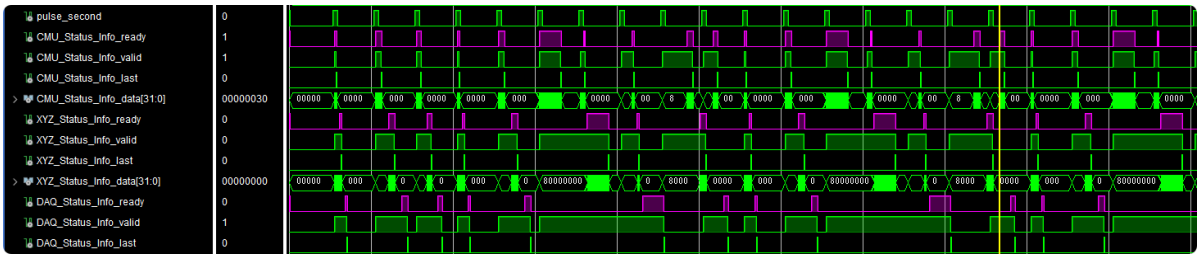
如图，前两个秒脉冲3个板子的输入数据速度小，能够正常发送。 第三个秒脉冲数据，数据输入大，当前模块（cmu）还没有发送完，当前模块的下一个帧的valid信号就已经拉高等待着ready了。因为当前模块优先级较高，会连着2次发送数据。当前模块的帧发送完成后，valid信号拉低，才能轮到下一个模块（xyz）发送速率。

总结一下就是cmu输入速率大把输出带宽占满了，其他模块发送要得等一等了。

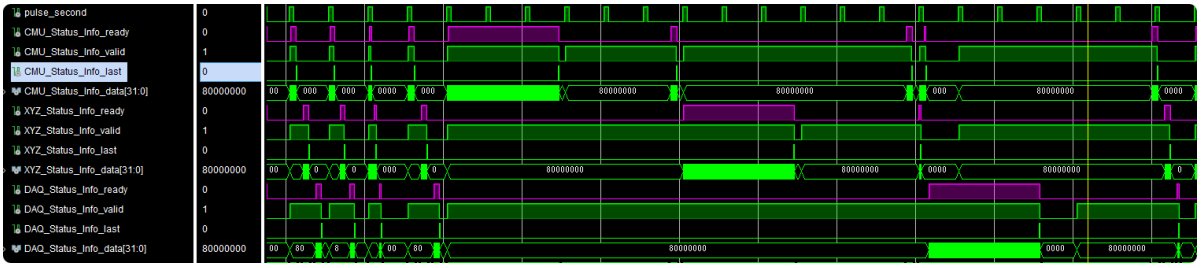


要解决这个问题，要么提高输出速率的带宽，要么减小输入数据速率，要么就花点时间多等等，反正最后数据是完整的，**只要不给ready的时候板子会等着，不会丢数据就行。**这边的输出速率指的是是写入send_fifo的数据速率(32bit x125MHz)，而读出位宽是8bit的，如果给125MHz，那实际发送速率就是8bit x125MHz，输出带宽局限于udp_send的fifo读出速率。

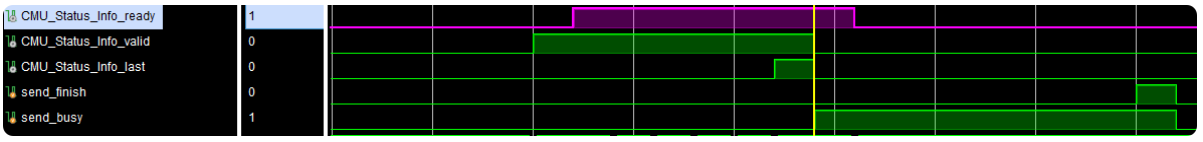
变换帧大小验证，正确。



偶尔来一个1400字节的巨帧，都没问题。只要cmu不一直把输出把带宽占满。



该数据处理模块是以帧为单位，接收一帧，转发一帧，一个帧中可以包含无数个transaction。因为需要给帧添加指令头494e,和crc32,所以send需要多花2个时钟。



综合

没有critical错误，解决多级驱动报错

Project name:	TCP_tp
Project location:	D:/shortname/UDP_Data_Process_VM/CMU_UDP_prj/TCP_tp
Product family:	Zynq UltraScale+
Project part:	xczu9eg-ffvb1156-2-i
Top module name:	Top1
Target language:	Verilog
Simulator language:	Mixed

Synthesis

Status:	✔ Complete
Messages:	! 2342 warnings
Active run:	synth_1
Part:	xczu9eg-ffvb1156-2-i
Strategy:	Vivado Synthesis Defaults
Report Strategy:	Vivado Synthesis Default Reports
Incremental synthesis:	None